Linear Quadratic Gaussian Controller Design Using a Graphical User Interface: Application to the Beam-Waveguide Antennas

E. Maneri[1] and W. Gawronski[2]

The linear quadratic Gaussian (LQG) design algorithms described in [2] and [5] have been used in the controller design of JPL's beam-waveguide [5] and 70-m [6] antennas. This algorithm significantly improves tracking precision in a windy environment. This article describes the graphical user interface (GUI) software for the design LQG controllers. It consists of two parts: the basic LQG design and the fine-tuning of the basic design using a constrained optimization algorithm. The presented GUI was developed to simplify the design process, to make the design process user-friendly, and to enable design of an LQG controller for one with a limited control engineering background. The user is asked to manipulate the GUI sliders and radio buttons to watch the antenna performance. Simple rules are given at the GUI display.

I. Introduction

The linear quadratic Gaussian (LQG) controllers, as designed for the beam-waveguide (BWG) antennas, show significantly improved performance over the ones initially implemented. However, the design of the LQG controllers for the BWG antennas is an iterative process that requires some experience in controller design. In order to make the process of the controller design simple and user-friendly, a controller design graphical user interface (GUI) was developed. This article describes the Matlab-based software that allows for the basic design and fine-tuning of the antenna LQG controllers. The approach is based on the algorithm described in [2] and [5], which was coded in Matlab and used in the LQG controller design of the BWG [5] and 70-m antennas [6]. The GUI presented here was developed to simplify the design process and to create a user-friendly environment in which one with a limited control engineering background may design an LQG controller.

II. Open-Loop Antenna Model

Prior to antenna controller design, one must collect information pertaining to the open-loop antenna model. The antenna open-loop (or rate-loop) model is obtained either from finite-element analysis or from system-identification procedures (that include field tests). The model consists of the antenna structure and the azimuth and elevation drives, as shown in Fig. 1.

[1] Student at Montana State University, Bozeman.

[2] Communications Ground Systems Section.

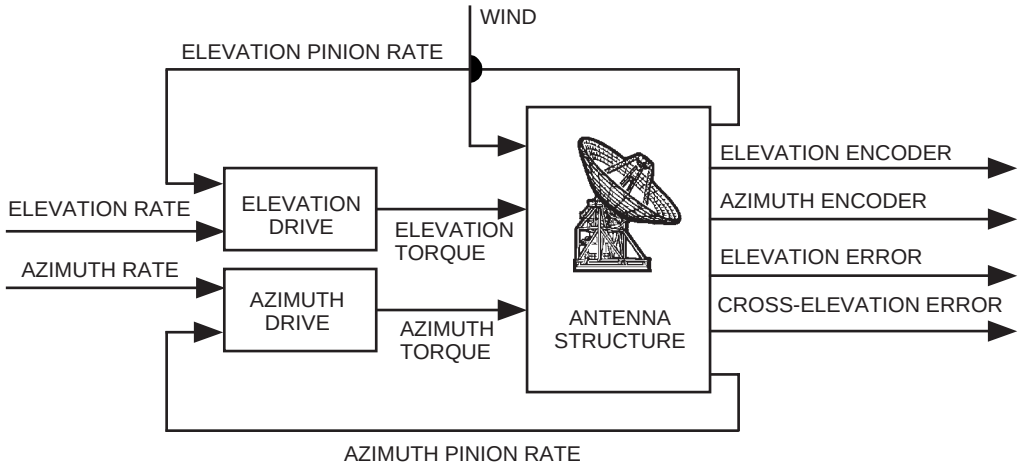


Fig. 1. The open-loop (or rate-loop) antenna model.

The inputs are azimuth and elevation rates and wind-gust disturbances. The outputs are azimuth- and elevation-encoder positions and cross-elevation and elevation beam errors. The rate-loop model in this form typically is obtained from a finite-element model. It is a two-axis model, in which wind gusts act directly on the structure surface.

The finite-element model can be simplified by decoupling the azimuth and elevation axes and consequently separating the azimuth and elevation models. This is justified by a weak cross-coupling between the azimuth command and elevation encoder and between the elevation command and azimuth encoder. Less than one one-thousandth of the coupling exists between the azimuth and elevation command/encoder than exists between the command and encoder readings of either one of these axes alone.

The rate-loop models obtained from field tests are one-axis models. For these models, cross-elevation and elevation beam-position outputs are unavailable, and the wind disturbances are applied to the rate input rather than to the antenna structure. A typical model obtained from the field data is shown in Fig. 2.

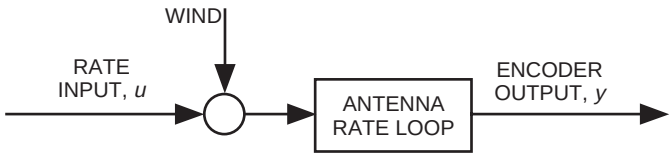


Fig.2. The simplified rate-loop model.

A. Model Equations

The rate-loop model is linear and is represented in the state–space form as the rate-loop triple, (A, B, C), where A is the state matrix, B is the input matrix, and C is the output matrix. It is described by the following linear constant coefficient differential equations:

$$\left.\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}\right\} \tag{1}$$

where the N-dimensional vector x is called the state vector, the s-dimensional vector u is the system input, and the r-dimensional vector y is the system output.

The state–space representation and the state vector are not unique, and one can introduce a new state variable, x_n, such that $x = Rx_n$. For a nonsingular transformation matrix R, a new state representation, (A_n, B_n, C_n), is obtained:

$$\left.\begin{aligned} A_n &= R^{-1}AR \\ B_n &= R^{-1}B \\ C_n &= CR \end{aligned}\right\} \tag{2}$$

In this representation, the input–output relationship is the same as in the original representation, (A, B, C). Although input–output relations remain invariant, it makes a difference for system analysis or for controller design which state or representation is chosen. Some representations have useful physical interpretations; others are more suitable for analysis and design.

Further, we will consider a modal representation. The modal state–space representation has a triple (A_m, B_m, C_m) characterized by the block-diagonal matrix, A_m, and the related input and output matrices:

$$\left.\begin{aligned} A_m &= \mathrm{diag}(A_{mi}) \\ B_m &= \begin{bmatrix} B_{m1} \\ B_{m2} \\ \vdots \\ B_{mn} \end{bmatrix} \\ C_m &= \begin{bmatrix} C_{m1} & C_{m2} & \cdots & C_{mn} \end{bmatrix} \end{aligned}\right\} \tag{3}$$

$i = 1, 2, \cdots, n$, where A_{mi}, B_{mi}, and C_{mi} are $2 \times 2, 2 \times s$, and $r \times 2$ blocks, respectively. The blocks A_{mi} are in the form

$$A_{mi} = \begin{bmatrix} 0 & \omega_i \\ -\omega_i & -2\zeta_i\omega_i \end{bmatrix} \tag{4}$$

where ω_i and ζ_i are the natural frequency and damping ratio of the ith mode. The state x_m of the modal representation consists of n independent components, x_{mi}, that represent a state of each mode:

$$x_m = \begin{Bmatrix} x_{m1} \\ x_{m2} \\ \vdots \\ x_{mn} \end{Bmatrix} \tag{5}$$

The state of the ith component is as follows:

$$x_{mi} = \begin{Bmatrix} q_{mi} \\ \dfrac{\dot{q}_{mi}}{\omega_i} \end{Bmatrix} \tag{6}$$

where q_{mi} and $\dot{q}_{mi}$ are the displacement and velocity of the ith mode.

From Eq. (3) and the block-diagonal form of the matrix A_m, it follows that the ith mode has the state–space representation, (A_{mi}, B_{mi}, C_{mi}), and is independently obtained from the state equations

$$\left.\begin{aligned} \dot{x}_{mi} &= A_{mi}x_{mi} + B_{mi}u \\ y_i &= C_{mi}x_{mi} \end{aligned}\right\} \tag{7}$$

such that $y = \sum_{i=1}^{n} y_i$.

B. Model Reduction

The order of the antenna model, obtained either from the finite-element analysis or from the system identification, is large: the finite-element model consists of thousands of degrees of freedom, while the model obtained from the system identification consists of many redundant states that represent the measurement noise. In both cases, the excessive order of the antenna model needs to be reduced in order to minimize the complexity of the controller. The model-order reduction presented in the following is carried out in modal coordinates, as described in [1].

A reduced-order model is obtained here by assigning value numbers to the modal states and truncating the least "valuable" ones. The states are evaluated using the H_∞ norm. The H_∞ norm of a system in modal coordinates, as in Eq. (7), is approximately given as

$$h_i \simeq \frac{\|B_{mi}\|_2 \|C_{mi}\|_2}{2\zeta_i \omega_i} \tag{8}$$

c.f., [1, Eq. (4.24a)]. Next, the modal state vector, x_m, of n modes (or $2n$ states) is reordered such that its components are of decreasing norm value: the norm h_1 of the first component, x_{m1}, is the largest one, while the norm h_n of the last component, x_{mn}, is the smallest one. Next, x_m is partitioned into two parts—retained and truncated:

$$x_m = \left\{ \begin{array}{c} x_{mr} \\ x_{mt} \end{array} \right\} \tag{9}$$

where x_{mr} is the vector of the retained states and x_{mt} is a vector of truncated states. Let the state triple (A_m, B_m, C_m) be partitioned accordingly:

$$\left.\begin{aligned} A_m &= \begin{bmatrix} A_{mr} & 0 \\ 0 & A_{mt} \end{bmatrix} \\ B_m &= \begin{bmatrix} B_{mr} \\ B_{mt} \end{bmatrix} \\ C_m &= \begin{bmatrix} C_{mr} & C_{mt} \end{bmatrix} \end{aligned}\right\} \tag{10}$$

If there are $k < n$ retained modes, x_{mr} is a vector of $2k$ states, and x_{mt} is a vector of $2(n-k)$ states. The reduced model is obtained by deleting the last $2(n-k)$ rows of A_m and B_m and the last $2(n-k)$ columns of A_m and C_m, obtaining the reduced-order model (A_{mr}, B_{mr}, C_{mr}).

Note that the modal reduction by truncation of stable models always produces a stable reduced model, since the poles of the reduced model are a subset of the full-order poles.

III. Performance Criteria

The antenna performance is characterized mainly by its step responses, rate-offset errors, wind-disturbance rms errors, and transfer-function bandwidth.

The step response is characterized by the settling time and the overshoot, the first of which is defined as the time at which the antenna-encoder output remains within 3 percent threshold of the nominal value of the step command. A 14.8-s settling time due to a unit step command is illustrated in Fig. 3(a). Overshoot (in percent) is the relative difference between the maximal encoder output and the commanded step with respect to the value of the commanded step. In Fig. 3(a), the overshoot is 18 percent.

The magnitude and phase of the transfer function of the DSS-26 antenna rate-loop model (azimuth axis) is shown in Fig. 4. The model has one pole at zero (the slope of the magnitude of the transfer function is −20 dB/decade for low frequencies). This pole is observed as a rigid-body rotation of the antenna with respect to the azimuth axis, and, in analytical terms, the system behaves as an integrator at low frequencies. The closed-loop transfer function of the antenna is shown in Fig. 3(b). Bandwidth is the frequency at which the magnitude drops 3 dB, or to 70.7 percent of its zero-frequency level (which is 1). This is illustrated in Fig. 3(b), where the bandwidth is 0.16 Hz.

The rate-offset response is characterized with the steady-state error. Fig. 5 shows the 0.06 deg/s rate offset command and the antenna response. The steady-state error (lagging) is 0.012 deg.

Wind-gust action is characterized by the rms errors of the encoders.

Although we do not plot rms error separately, its value is computed in the simulation. For a visual display of an antenna-encoder response to 32-km/h wind gusts, see Fig. 6, where there is an rms value of 0.10 mdeg.

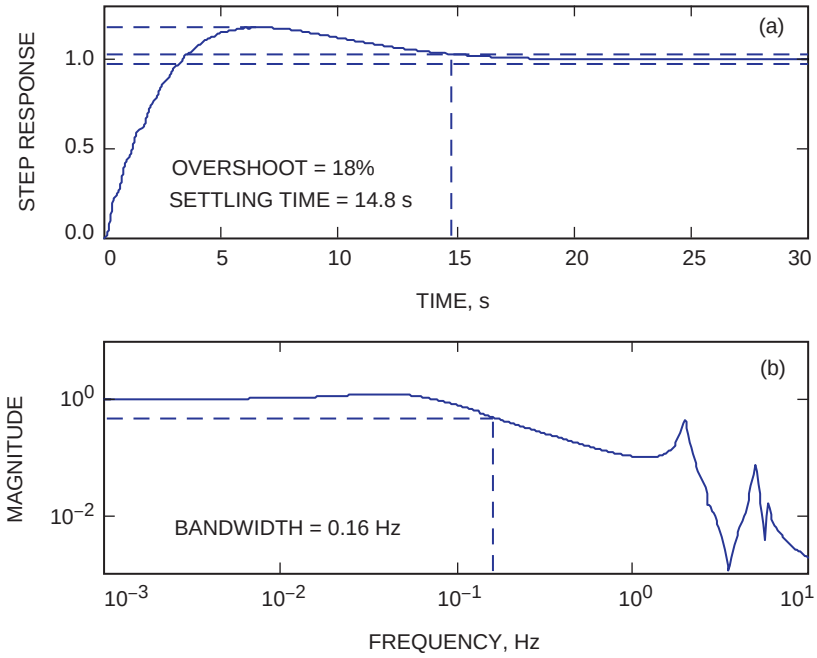


Fig. 3. The antenna (a) step-response overshoot and settling time and (b) transfer-function bandwidth.

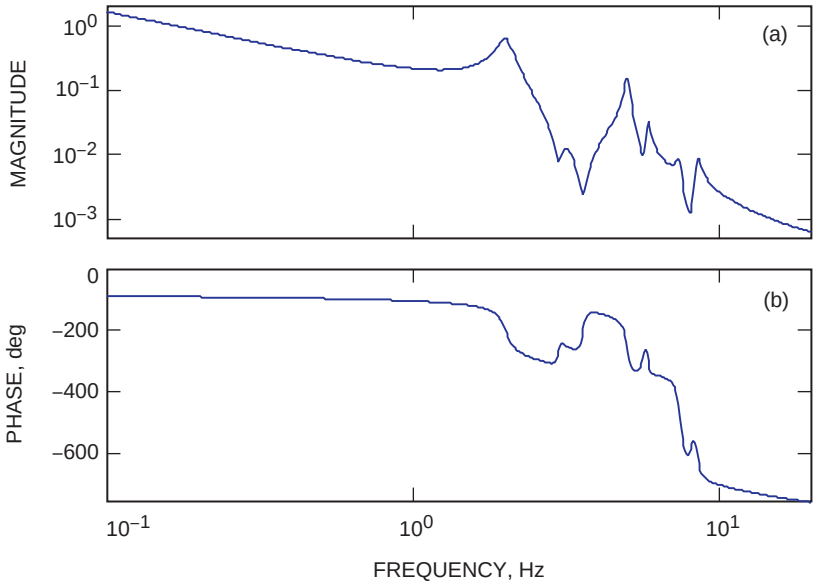


Fig. 4. Bode plots of the DSS-26 azimuth rate-loop model: (a) magnitude and (b) phase.

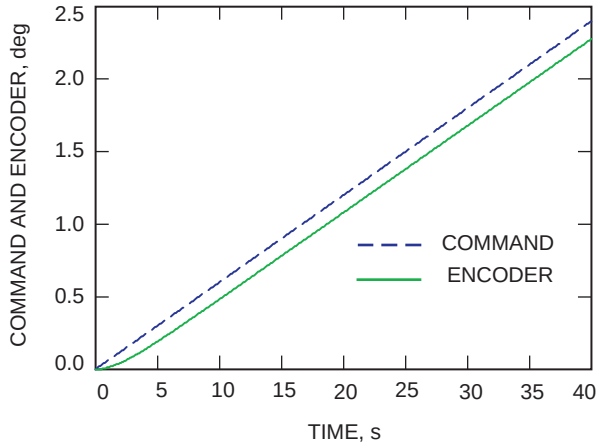


Fig. 5. The rate-offset command and the antenna-encoder reading.

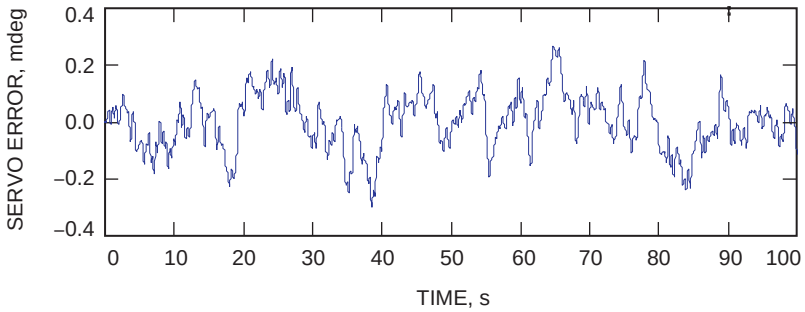


Fig. 6. The antenna servo error in 32-km/h wind gusts.

IV. Designing the LQG Controller

The controller design process consists of choosing the controller configuration and determining the controller gains.

A. Controller Configuration

The LQG controller consists of the estimator, the PI controller, and the flexible-mode controller (see Fig. 7). The detail of each part is shown in Fig. 8 and is described as follows.

First, the antenna state–space model (A, B, C) also includes the disturbances, v, and the measurement noise, w:

$$\dot{x} = Ax + Bu + v \tag{11a}$$

$$y = Cx + w \tag{11b}$$

(see Fig. 8). (In the preceding and in the following, the subscript m is omitted for convenience of notation). The disturbances (predominantly wind gusts) have covariance V. The measurement noise has covariance W. It is assumed additionally that the input and output noises are not correlated. This assumption is equivalent to independence of their sources. Indeed, the measurement noise is independent of the wind disturbances.

The purpose of the estimator is to evaluate the antenna dynamic states using the rate input, u, and the encoder output, y, of the antenna. The estimated state vector is denoted $\hat{x}$, and the error between the actual encoder output and the estimated output is defined as $\varepsilon = y - C\hat{x} = y - \hat{y}$. The estimated state is obtained from the following equation:

$$\dot{\hat{x}} = A\hat{x} + Bu + K_e\varepsilon \tag{12}$$

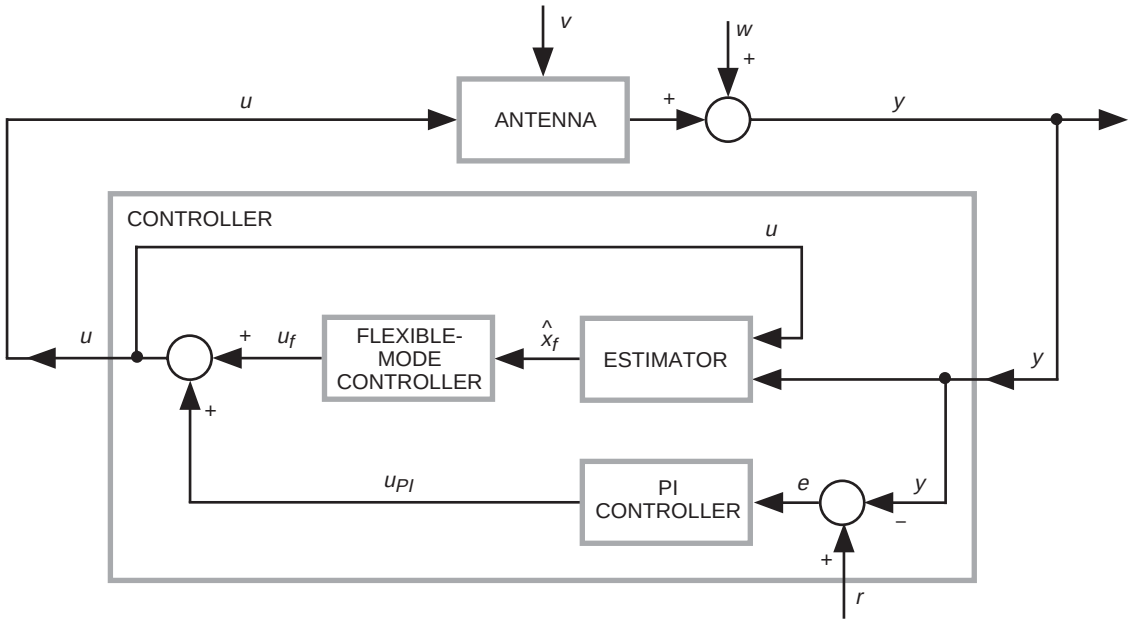
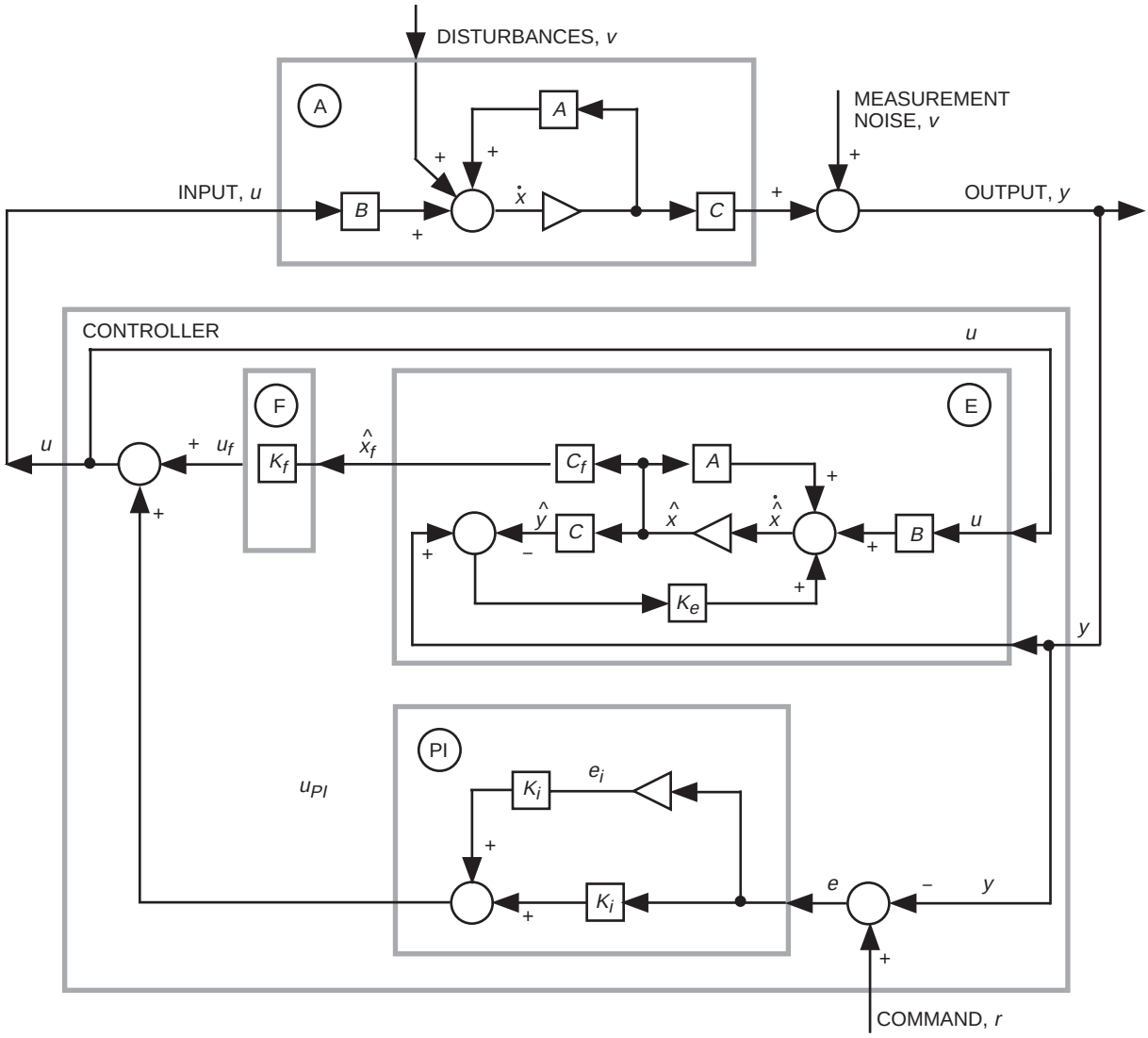


Fig. 7. The structure of the LQG control system.



VARIABLES	
y	= ENCODER OUTPUT
u	= ANTENNA RATE INPUT
r	= COMMAND
e	= SERVO ERROR
e_i	= INTEGRAL OF THE SERVO ERROR
x	= ANTENNA STATE
$\hat{x}$	= ESTIMATED ANTENNA STATE
ε	= ESTIMATION ERROR
v	= DISTURBANCES
w	= MEASUREMENT NOISE

BLOCKS	
K_e	= ESTIMATOR GAIN
k_p	= PROPORTIONAL GAIN
k_i	= INTEGRAL GAIN
K_f	= FLEXIBLE-MODE GAIN
A	= STATE MATRIX
B	= INPUT MATRIX
C	= OUTPUT MATRIX
C_f	= FLEXIBLE-MODE OUTPUT MATRIX

Fig. 8. The detailed structure of the LQG controller: A = antenna (hardware); E = estimator (software); PI = PI controller (software); and F = flexible-mode controller (software). A triangle denotes integration.

One can see that the estimated state is proportional to the estimation error; the proportionality gain, K_e, is called the estimator gain.

The controller forms the negative feedback between the estimated state, $\hat{x}$, and the antenna input, u:

$$u = -K_c\hat{x} \tag{13}$$

where the controller gain matrix is K_c.

The task is to determine both the controller and estimator gains such that the performance index, J, is minimal, where

$$J^2 = E\left(\int_0^\infty \left(x^T Q x + u^T R u\right) dt\right) \tag{14}$$

and Q and R are user-defined weight matrices, Q is a positive semi-definite matrix, and R is a positive definite matrix.

It is known (see [3,4]) that the minimum of J obtained for the controller-gain matrix is obtained as

$$K_c = R^{-1}B^T S_c \tag{15}$$

where B is the input matrix of the antenna model and S_c is a solution of the following Riccati equation:

$$A^T S_c + S_c A - S_c B R^{-1} B^T S_c + Q = 0 \tag{16}$$

Similarly, the optimal gain in the estimator equation, Eq. (12), is given as follows:

$$K_e = S_e C^T W^{-1} \tag{17}$$

where C is the output matrix of the antenna model and S_e is the solution of the following algebraic Riccati equation:

$$A S_e + S_e A^T - S_e C^T W^{-1} C S_e + V = 0 \tag{18}$$

Finally, the controller gain is split into proportional, integral, and flexible-mode gains,

$$K_c = \lfloor\, k_i \quad k_p \quad K_f \,\rfloor \tag{19}$$

to fit into the configuration as in Fig. 7.

The preceding presentation shows that the LQG controller is a model-based controller, where the antenna model is a part of the controller. It is used to estimate the non-measured antenna states. In order to ensure estimation accuracy, the antenna model shall closely match the actual antenna dynamics (thus, the finite-element model is unacceptable). Generally, the finite-element model is used in the antenna

design stage when a model of a "real" antenna is not yet available. For implementation purposes, the antenna model is obtained by conducting field tests and system identification.

It also was shown that the states of the LQG controller are divided into the tracking states (antenna-position error and its integral) and the flexible-mode states (flexible-mode displacements). For this configuration, the antenna-state vector is of the following form:

$$x = \{\, e_i \quad e \quad x_a \quad x_{f11} \quad x_{f12} \quad \cdots \quad x_{fn1} \quad x_{fn2} \,\}^T \tag{20}$$

where e_i is the integral of the servo error, e is the servo error, and x_a is a variable from the system-identification model (of unexplained physical interpretation). Each flexible mode is described by two state variables; thus, x_{f11} and x_{f12} are the two state components of the first natural mode, and x_{fn1} and x_{fn2} are the two state components of the nth natural mode. The BWG antennas typically have $n = 3, 4$, or 5 natural modes.

B. Determining the LQG Controller's Gains by Tuning Its Weights

The LQG controller is shown in Fig. 8. It consists of an estimator that estimates the antenna states based on measured antenna input, u, and output, y. The estimator has its own gain, K_e, to ensure the minimal estimation error. The proportional, k_p, integral, k_i, and flexible-mode, K_f, gains also are included in the model inputs to control the estimated state and the output. The design task is to determine the gains such that the antenna performance is optimal.

We explain the process of determining the LQG gains, given in [2], in a heuristic way. The controller gains depend on the weight matrix, Q, covariance matrix, V, and scalars R and W. First, one picks values of the latter scalars: assume unity. Next, one picks the weight matrix, Q. It shapes the optimization index, a positive variable to be minimized. The covariance matrix specifies the noise in the system, and it impacts the estimator gains. The difficulty arises when one tries to express the performance of the antenna (such as the rms servo error in wind, closed-loop bandwidth, etc.) through the values of Q and V. A closed-form relationship does not exist between those matrices and the required behavior of an antenna. Thus, an ad hoc solution is a trial-and-error approach. However, there are too many parameters in Q and V to make this approach effective. Moreover, Q and V depend on the choice of the state–space coordinates, with some coordinates more useful than others. Physical coordinates, such as structural displacements, motor torques, or currents, although easy to interpret, are highly coupled, and therefore create undesirable difficulties. It was verified that the modal coordinates simplify the design because they are weakly coupled (i.e., modifications of one of them weakly impact the remaining ones). In consequence, Q and V matrices are diagonal (therefore, there are fewer parameters to control the closed-loop dynamics). Additionally, we assume the equality of Q and V, i.e., $Q = V$. This simplification eases the search for the "best" controller and is rather opportunistic, with a goal to simplify the GUI approach. Experience shows, however, that the results obtained using this assumption are satisfactory, or even exceed expectations. The reader shall note that, in individual cases, the tuning of Q and V separately may improve the performance.

Taking into account the above considerations, the matrices Q and V are diagonal, i.e., $Q = V = \mathrm{diag}(q)$, where q is a weighting vector. Note that components of q correspond to the components of the state vector, x. Thus, for the state vector, as in Eq. (20), the weighting vector has the form similar to x:

$$q = \{\, q_{ei} \quad q_e \quad q_a \quad q_{f1} \quad q_{f1} \quad \cdots \quad q_{fn} \quad q_{fn} \,\}^T \tag{21}$$

where q_{ei} is the weight of the integral of the servo error, q_e is the weight of the servo error, q_a is the weight of the variable x_a, q_{f1} is the weight of the first natural mode, and q_{fn} is the weight of the nth natural mode.

Now, we have to specify $2n + 3$ weights, which for a typical number, $n = 4$, makes 11 weights. The critical fact is that the weights have weak dependence amongst each other: the ith modal weight, q_{fi}, impacts mostly the ith mode, i.e., state variables, x_{fi1} and x_{fi2}. Also, the tracking weights, q_e, q_{ei}, and q_a, impact mostly the tracking states, e, e_i, and x_a (see [2]). The design process is illustrated in the next subsection.

C. GUI for the LQG Controller Design

The LQG controller-design GUI tool was developed to make the process of controller design user-friendly. The GUI displays are shown in Fig. 9 and discussed in the following.

On the left side of the interface in Fig. 9, there is a short description of the tool. The description falls under the heading "LQG Controller for BWG Antenna." The linear quadratic Gaussian (LQG) controller is a model-based controller, meaning the antenna's linear model is a part of the controller. Quadratic refers to the index that the controller minimizes, and Gaussian refers to disturbances and noises acting on the antenna. The GUI allows for simple manipulations of the design parameters and observations of the antenna performance to make design decisions. Alternative methods involve code manipulation.

The frame below the description contains a ".mat" file (Matlab's data file). In this case, this file holds the A, B, C and D matrices [see Eqs. (11a) and (11b)]. These are the parameters of the antenna rate-loop model: a is a square matrix, n-by-n; b is a column vector, n-by-1; c is a row vector, 1-by-n; and d is a scalar. The user must enter "load filename.mat" in the editable text box.

We now focus on the eight sliders and their three displays. These are the user tools for modifying the controller's performance. (To understand how these play a role in controller design, see Subsection IV.B). Each slider (except the wind-speed slider) ranges from 0 to 100 (these are dimensionless numbers). By

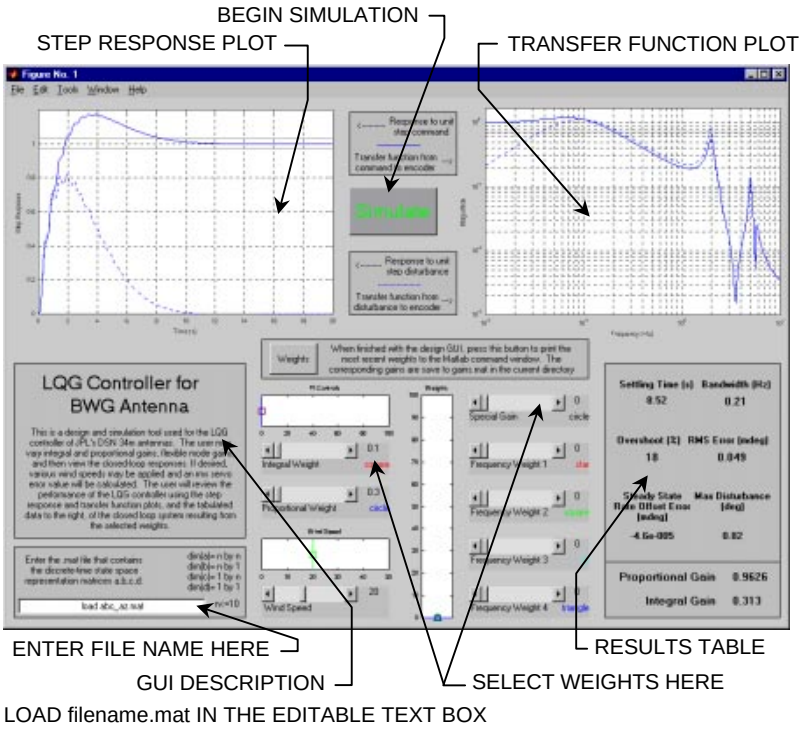


Fig. 9. The interface of the LQG design.

clicking on the end arrows, the marker will move one tenth of a slider unit in that direction. Clicking in the slider's path will move it one slider unit in that direction. Alternatively, the user may drag the marker any distance, but this generally is not recommended because large weight variations may result in an unstable antenna. Additionally, if the user randomly moves the slider, it will be difficult to attain similar weight values at a later time.

To be sure of what value the controller is using, note the numerical value directly to the right of the slider and the appropriate display of the three adjacent displays (for relative weight comparisons). The method of determining where to position the marker along each slider is documented later in Section IV.D. Also, see the tool tip strings (tool tip strings are the little yellow messages that pop up when the mouse lingers too long over one spot) for a description of which slider effects what most heavily. If one chooses to proceed without understanding the cause and effect relationships of this controller, the best advice is to start with all the weights at zero and to *slowly* adjust them.

In the bottom right corner of the GUI screen are the simulation results and the values of the proportional and integral gains. Each of the result headings will display a one-sentence description of what it represents in a tool tip string. Each of the values below is updated after the "simulate" button is pressed.

The upper-left plot is the step response to a 1-deg offset. The upper-right plot is the magnitude of the transfer function of the closed-loop system. The axes are labeled, and between the two plots there is a legend.

The "simulate" button is used to execute the simulation with the parameters on the screen.

The antenna performance can be observed on the GUI display (Fig. 10 is referenced):

(1) *Settling time* refers to the time it takes for the step response to get (and stay) within 3 percent of the commanded response. The two gray lines in the upper plot represent this ± 3 percent range. In the event that the user creates an unstable controller, this value will refer her/him to the Matlab command window (which must be open anyway), where a message describing the model's instability will be printed to the screen.

(2) *Bandwidth* is the frequency at which the magnitude of the transfer function moves below -3 dB (or below 0.71). The 0.71 value is marked in gray on the lower plot.

(3) *Overshoot* is the percent that the antenna overshoots its commanded value. Note that, if the antenna is unstable, the algorithm may give nonsense answers. One should do a visual check to verify this number.

(4) *Root-mean square (rms)* error indicates the servo-error value in wind gusts for the selected wind speed.

(5) Steady state *rate-offset error* is the difference between the actual antenna position and the commanded position when the antenna is moving with a constant rate. This value is zero under steady-state conditions.

(6) *Maximum disturbance* is the maximum value of the antenna response to the unit step disturbance.

In Fig. 9, below these six parameters, the proportional and integral gains are displayed, which are related, non-linearly, to the proportional and integral weights. These values are of importance because from them the experienced user may gather information about the controller's robustness.

The performance of the antenna depends on *all* controller gains, or *all* controller weights. The relationship is not simple and depends on the coordinates in which the antenna model is represented.

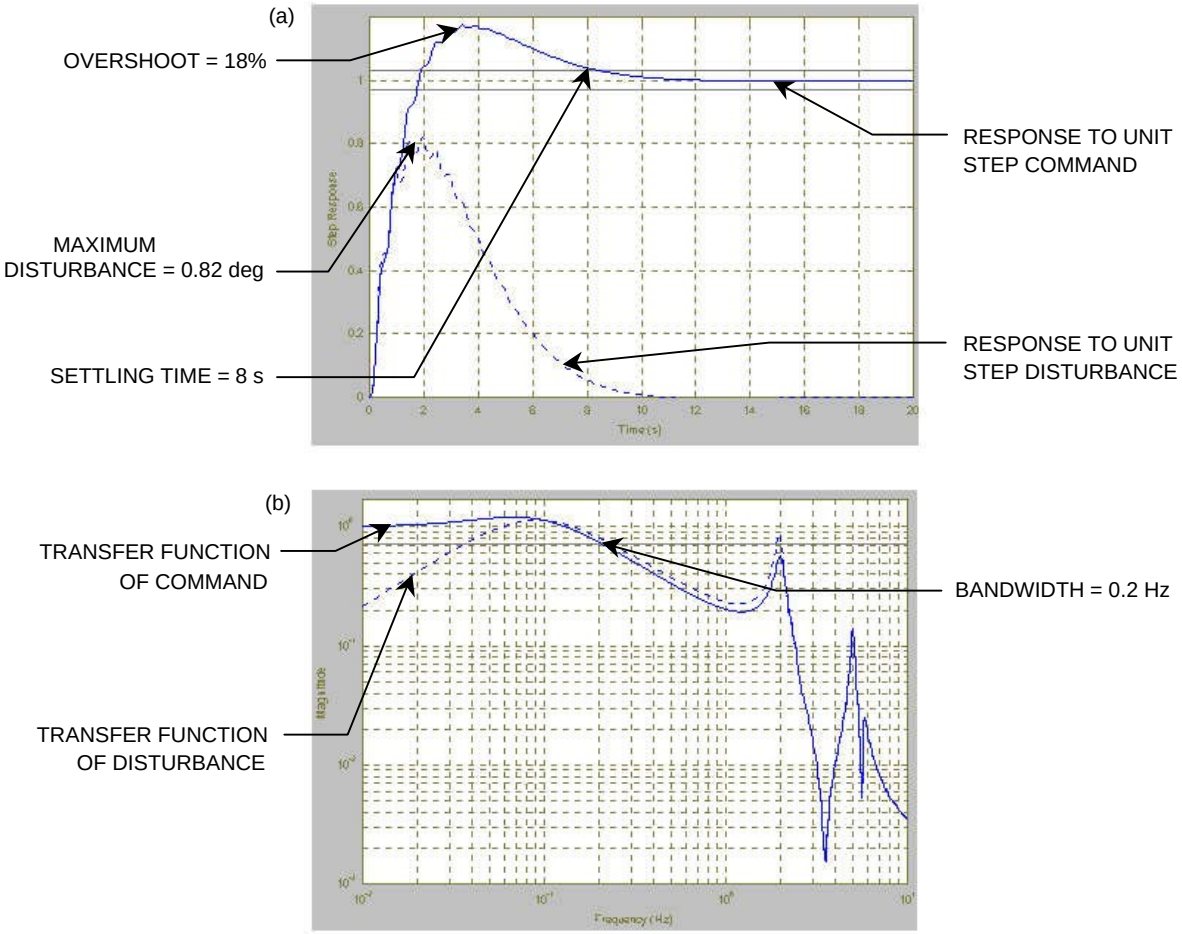


Fig. 10. Design GUI displays: (a) step responses and (b) magnitudes of the transfer functions.

Generally, changing a single weight impacts more than one performance feature. If one wants to change a single feature, e.g., expand the bandwidth, one needs to change many weights. However, in the *modal* coordinates of the antenna model, each weight influences predominantly one or two performance features and is weakly coupled with the remaining ones. This helps to follow the improvement of the antenna performance, since each slider predominantly addresses a single performance feature (such as bandwidth, vibrations, overshoot, etc.). Table 1 summarizes these relationships.

Figure 11 shows the relationships between the step response and the magnitude of the transfer function.

D. DSS-26 Antenna Example

The following steps (illustrated in Figs. 12 through 17) can be followed as a guideline to the controller design process. They reflect common antenna features.

(1) The design starts with the input of the state–space representation of the open-loop antenna model. For example, the DSS-26 antenna representation (A, B, C, D) is a file called "abc.mat" located at the directory "c:/matlabr11/lqg/gui/az." In the lower-left window, one enters "load c:/matlabr11/lqg/gui/az/abc.mat."

(2) Next (Fig. 12), one selects arbitrary, but small, values of proportional and integral weights (say, $q_{ei} = 0.1$ and $q_e = 0.3$) and zero for special and frequency weights. After simulation,

Table 1. Relationship between LQG weights and performance.

Slider movement (LQG weight)	Result (performance)
Proportional weight	Settling time, bandwidth, disturbance rejection
Integral weight	Overshoot, rate-offset error, disturbance rejection
Special weight	Moved in emergency only
Frequency weight 1	Vibration amplitude of the lowest (fundamental) frequency, height of the first resonance peak[a]
Frequency weight 2	Vibration amplitude of the second frequency, height of the second resonance peak[a]
Frequency weight 3	Vibration amplitude of the third frequency, height of the third resonance peak
Frequency weight 4	Vibration amplitude of the fourth frequency, height of the fourth resonance peak

[a] While switching from the azimuth to the elevation open-loop model, frequency weight 1 corresponds to the vibration amplitude of the second frequency, and frequency weight 2 corresponds to the vibration amplitude of the lowest (fundamental) frequency.

the step response shows excessive settling time and overshoot (8.52 s and 18 percent, respectively), visible flexible oscillations, and an unacceptable maximal disturbance step value (0.82 deg). The magnitude of the transfer function shows low bandwidth (0.21 Hz), sharp resonance peaks, and a larger-than-desired magnitude of the disturbance transfer function.

(3) The most excessive oscillations come from the first (or fundamental) mode. Therefore, the weight of the first flexible mode is increased to 5 (see Fig. 13), and other weights are unchanged. The results are visible in the step responses, where the oscillations are now invisible, and in the transfer function, where the first resonant peak disappeared. Other parameters remained unchanged except for the disturbance responses that deteriorated, and the second resonance peak is excessive.

(4) The proportional weight is increased to 10 (Fig. 14). This move reduces the settling time to 3.8 s, overshoot to 3.6 percent, and expands the bandwidth to 0.6 Hz. The maximum disturbance step response is lowered to 0.49 deg, but the second resonance peak has raised to a dangerous level.

(5) The integral gain is increased to 3 (Fig. 15). While bandwidth expands to 0.8 Hz and the maximum of the disturbance step response is reduced to 0.41, overshoot and settling time increase to 13 percent and 4.0 s, respectively.

(6) In this step (Fig. 16), the second resonance peak is damped by increasing the "frequency weight 3" to 10. The step response is smoother, and the second resonance peak in the magnitude of the transfer functions is reduced.

(7) In the final step, the proportional weight is increased to 25 and integral weight is increased to 12 (see Fig. 17). The settling time decreases (3.0 s), but overshoot increases (14 percent). The maximum step disturbance response decreases to 0.38 deg, and the bandwidth significantly increases to 2.2 Hz. This is an acceptable solution, and in order to save the weights governing the output, the "weights" button is pushed and the values are printed to the Matlab command window. Now the design window is closed and the next stage of the design begins—the fine-tuning procedure.

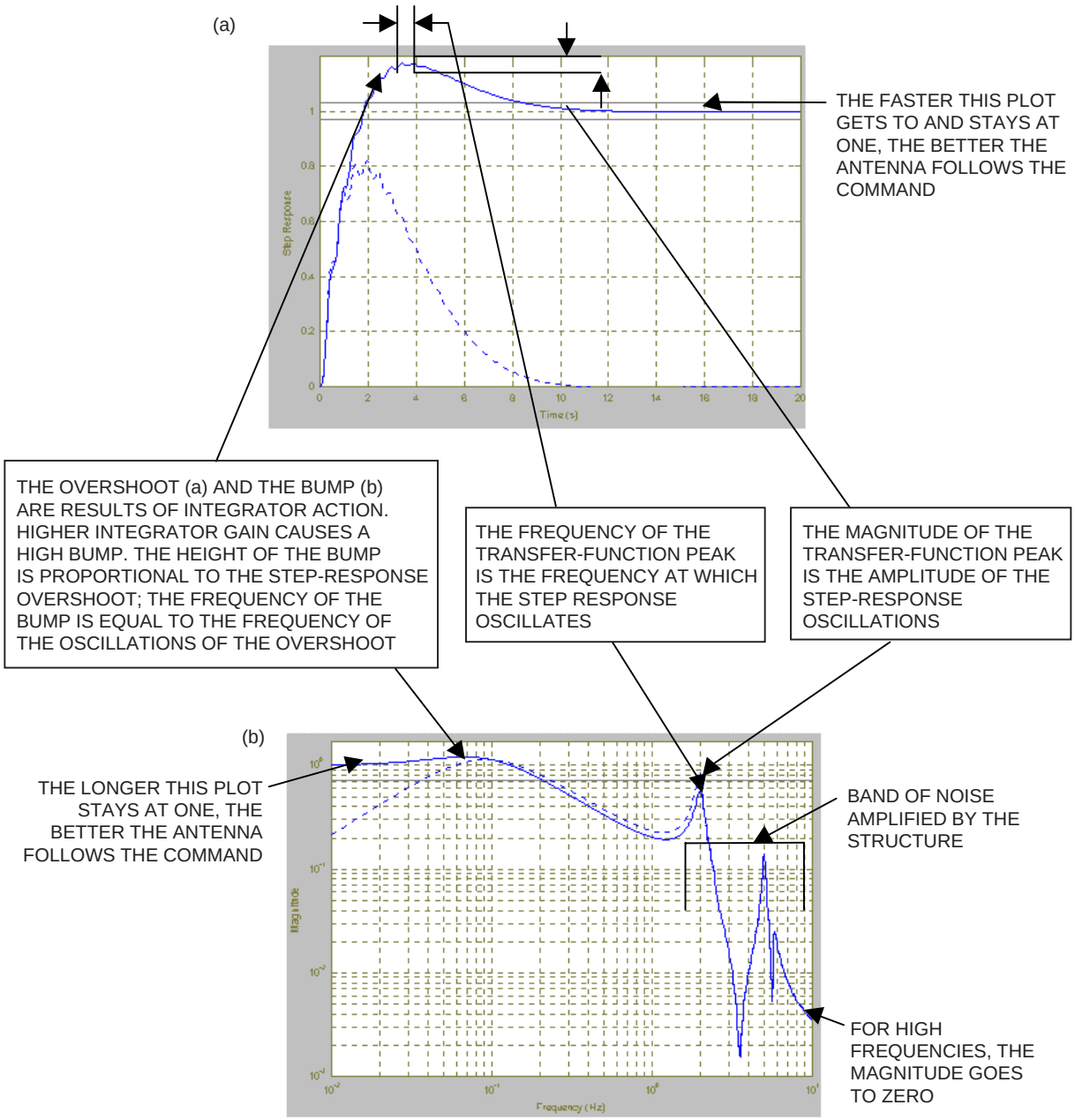


Fig. 11. Properties of the (a) step response and (b) magnitude of the transfer function.

V. Fine Tuning of the LQG Controller

Certain features of the controller obtained through the approach described above can be improved by further modifying the controller weights. For example, the oscillations of the closed-loop response can be further reduced or the closed-loop response can be modified such that the wind disturbance does not exceed its specification. However, the closed-loop states are not as easily decoupled as the open-loop states; therefore, the relationship between the weights and the closed-loop dynamics is less clear and often is difficult to track. For this reason, a constrained-optimization approach is used to tune the already designed controller.

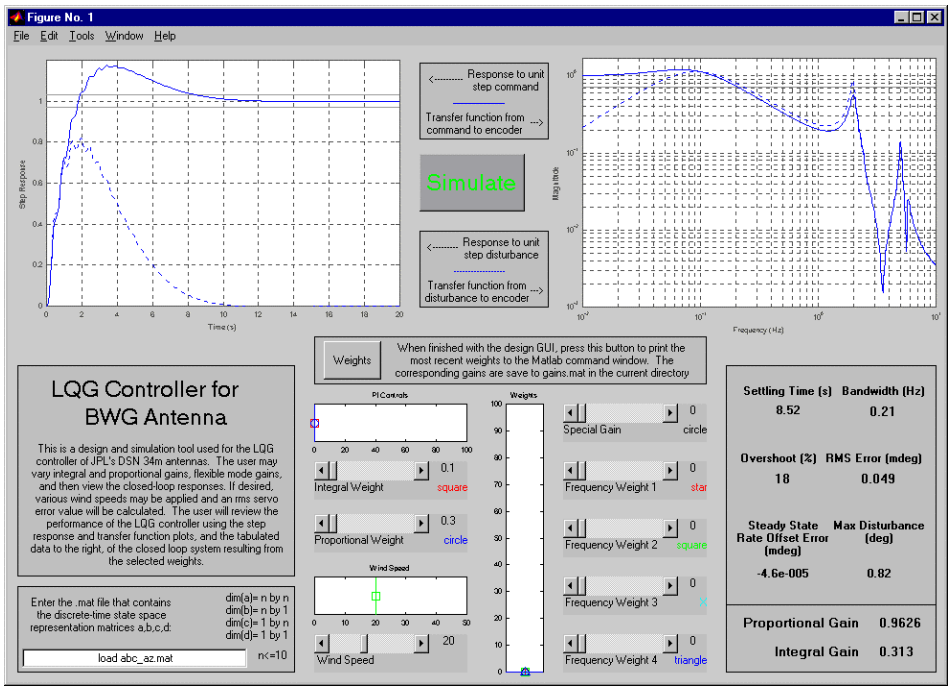


Fig. 12. Design GUI interface 1.

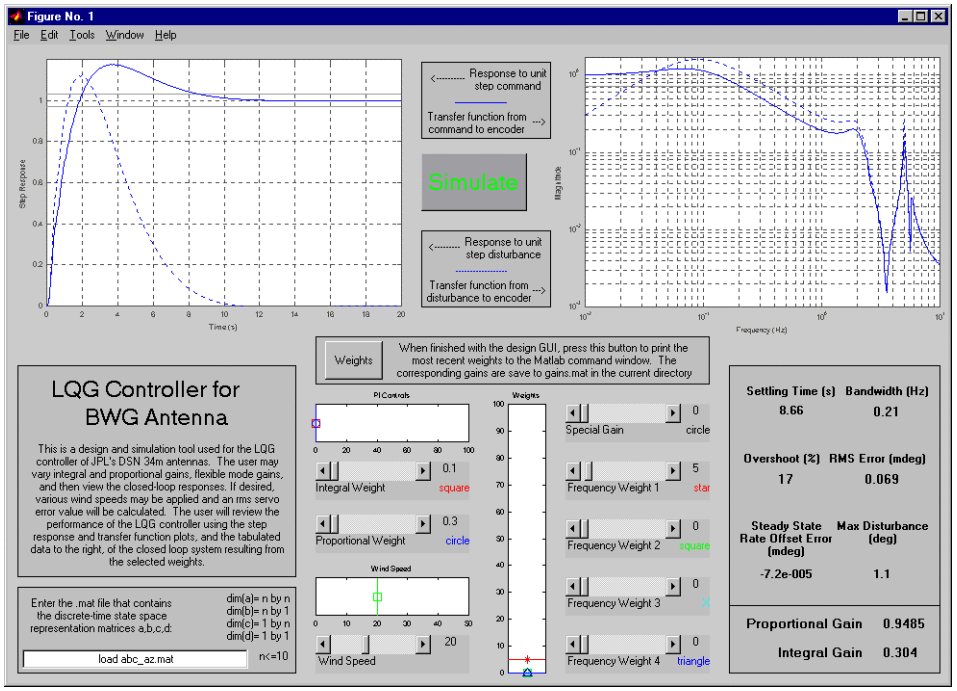


Fig. 13. Design GUI interface 2.

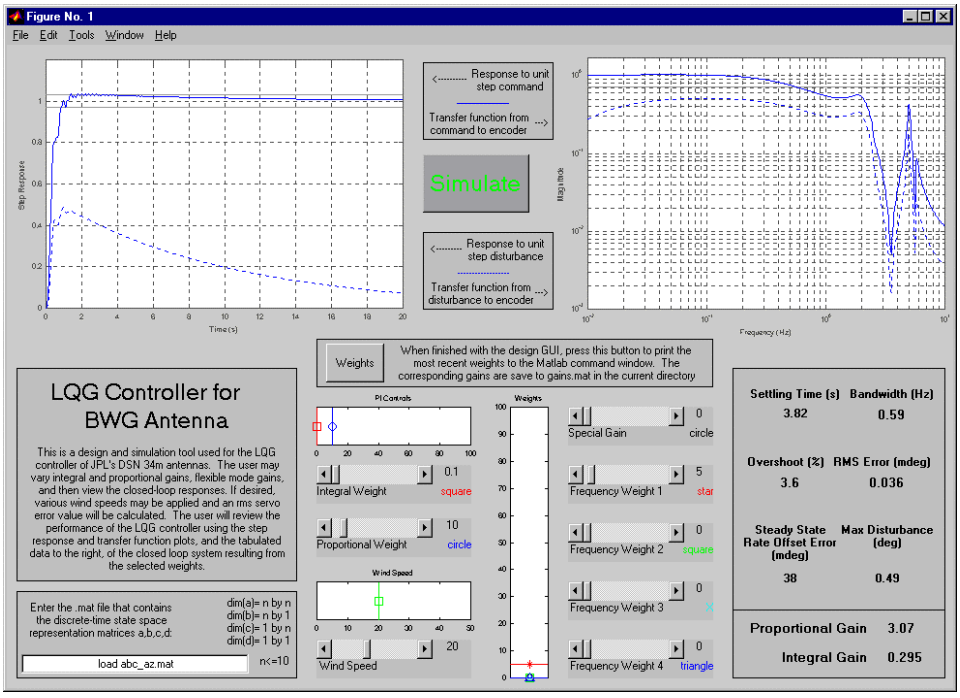


Fig. 14. Design GUI interface 3.

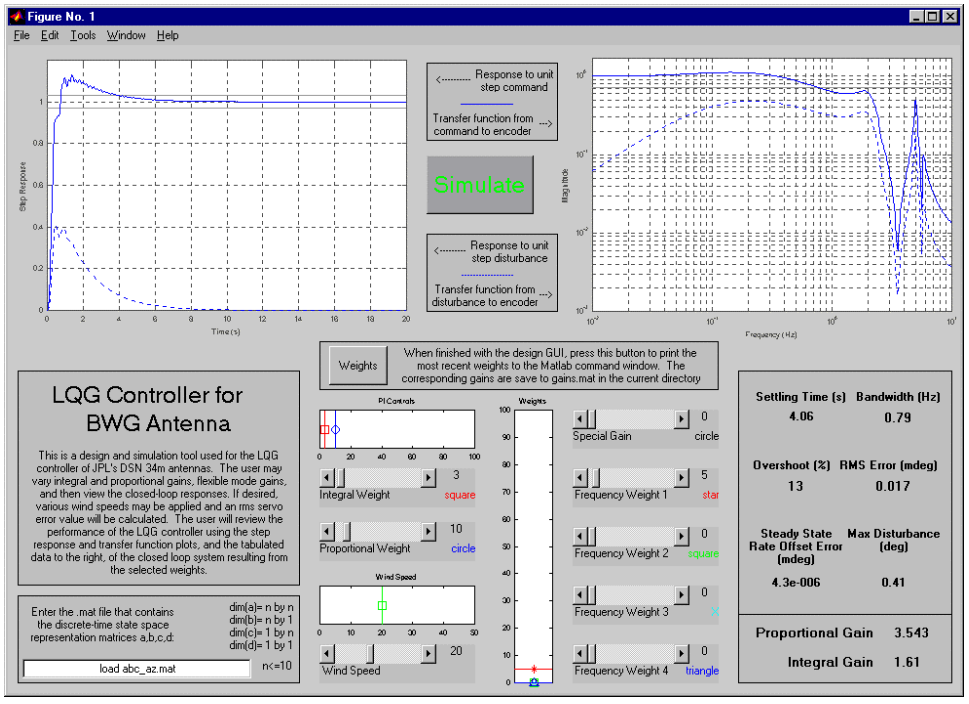


Fig. 15. Design GUI interface 4.

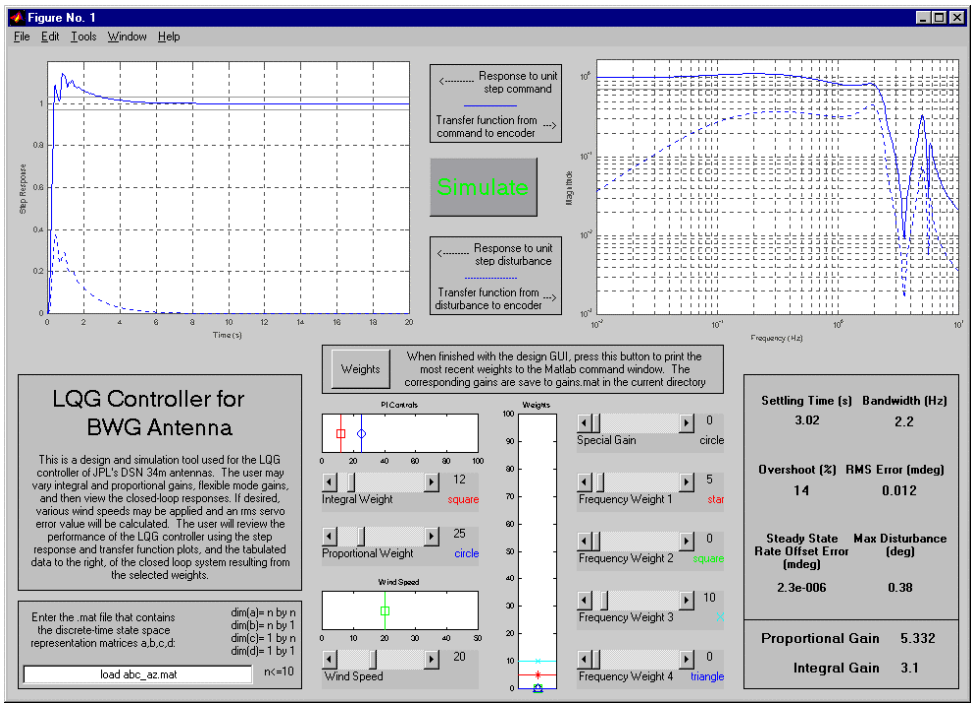


Fig. 16. Design GUI interface 5.

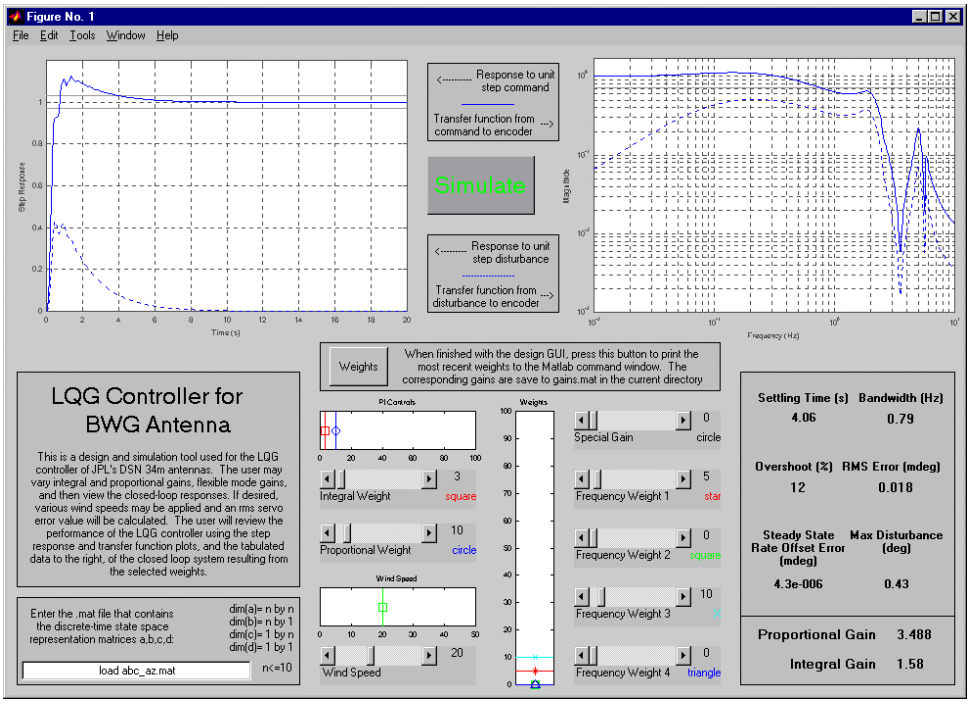


Fig. 17. Design GUI interface 6.

A. Description of Variables

In this approach, the weights are design parameters, and the following variables are either optimized or constrained:

$e_r =$ steady-state servo error in the 0.1 deg/s rate offset, mdeg, defined as

$$e_r = \|r(t) - y(t)\|_2 \tag{22}$$

for $18 < t < 20$ s

$e_{dmaz} =$ maximum value of the servo error due to the unit step disturbance, deg:

$$e_{dmaz} = \max(|e_d|) \tag{23}$$

for $t \geq 0$

$e_w =$ rms servo error in 24-km/h wind gusts, mdeg, for $0 \leq t \leq 100$ s

$e_o =$ overshoot of the unit step response, percent

$t_s =$ settling time of the unit step response, s

$f_o =$ bandwidth, Hz

$m_h =$ magnitude of the closed-loop transfer function (from the command to the encoder) for a high-frequency range (above 3.5 Hz):

$$m_h = \max\ m(f)$$

for $f \geq 3.5$ Hz

$m_{d\,\max} =$ magnitude of the closed-loop transfer function (from the disturbance to the encoder), i.e.,

$$m_{d\,\max} = \max\ m_d$$

for $f \geq 0$ Hz

The index (a positive function to be minimized) is defined as follows:

$$J_o = w_1 e_r + w_2 e_{d\,\max} + w_3 e_w + w_4 e_o + w_5 t_s + w_6(3 - f_o) + w_7 m_h + w_8 m_{d\,\max} \tag{24}$$

Each variable in the function to be minimized is weighted (or multiplied by a positive number). The weight w_i indicates the relative importance of each variable. If $w_i = 0$, the ith variable is not optimized. In the above-defined function, the bandwidth is maximized by using $3 - f_o$ variable rather than the bandwidth f_o itself. For BWG antennas, the bandwidth will not exceed 3 Hz; therefore, the minimization of $3 - f_o$ leads to expansion of f_o.

The variables above are functions of the LQG weights, $q_i, i = 1, \cdots, n$; therefore, J_o is a function of q as well. The initial values of the LQG weights are taken from the design part of the GUI, as described in Section IV.

B. GUI Description

This tool (shown in Fig. 18) is quite similar to the previously described tool. The two plots of the fine-tuning GUI are identical to those of the design GUI except that the step plot has a 10-s span of the x-axis in the fine-tuning GUI. The user is given a line on which to enter the file containing the discrete-time state–space representation of the rate-loop model (either azimuth or elevation axis).

In this GUI, there are eight variables (given above) that can be either minimized or constrained. Different variables and their changes are displayed and can be tracked from the start throughout. The user also may view the initial variable values, the iteration number, and the function value in the Matlab command window (see Fig. 19).

The fine-tuning GUI is meant only as a small improvement tool. This restriction follows from the fact that the program searches for local minimum only; hence, dramatic improvement is not expected, unless the initial design is a poor one. The requirement to dramatically improve the design (by setting demanding constraints, for example) usually leads to the termination of the program with the message "no feasible solution exists" in the Matlab command window.

The optimization function "constr" is used here to find a local minimum of a constrained, nonlinear, multivariable function. Before attempting to minimize a function, it will ensure that the constraints have been met. Two main decisions must be made when using the "constr" function. One is whether to optimize or constrain a certain variable. The other is how many iterations are necessary to run in a block. Too many iterations may require a long simulation time, perhaps causing the user to feel the process is out of control, and too few requires frequent restarting.

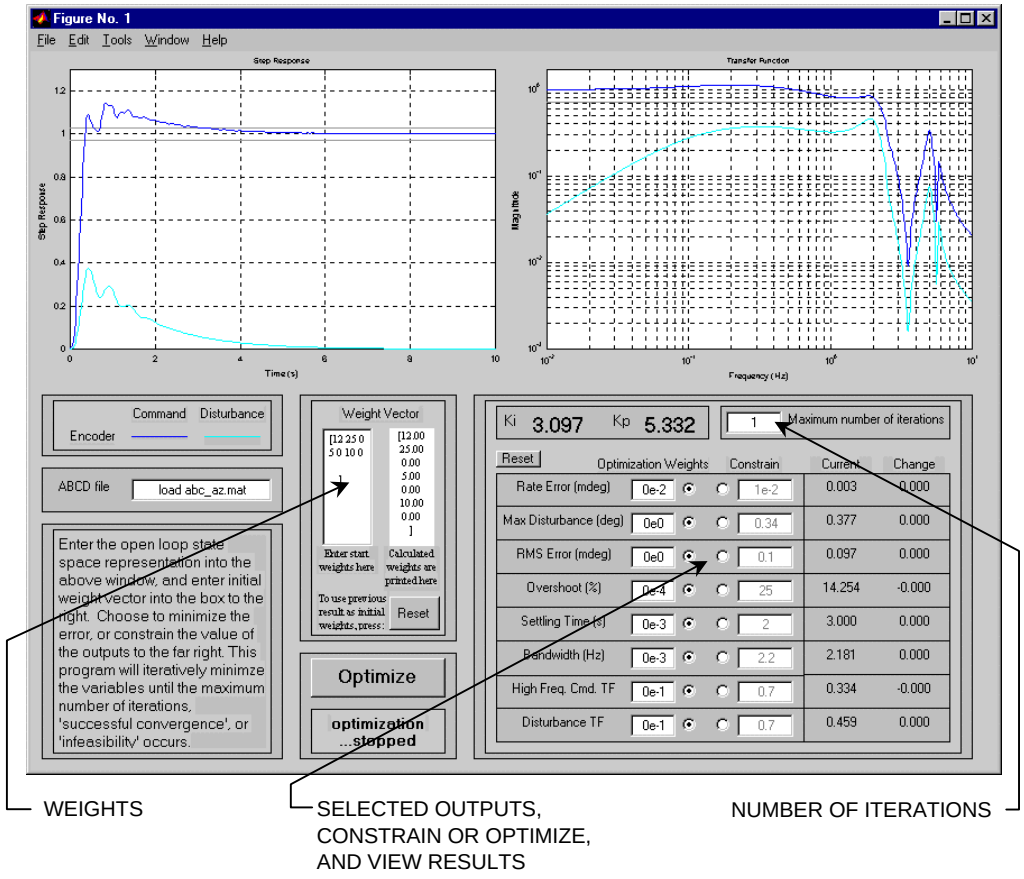
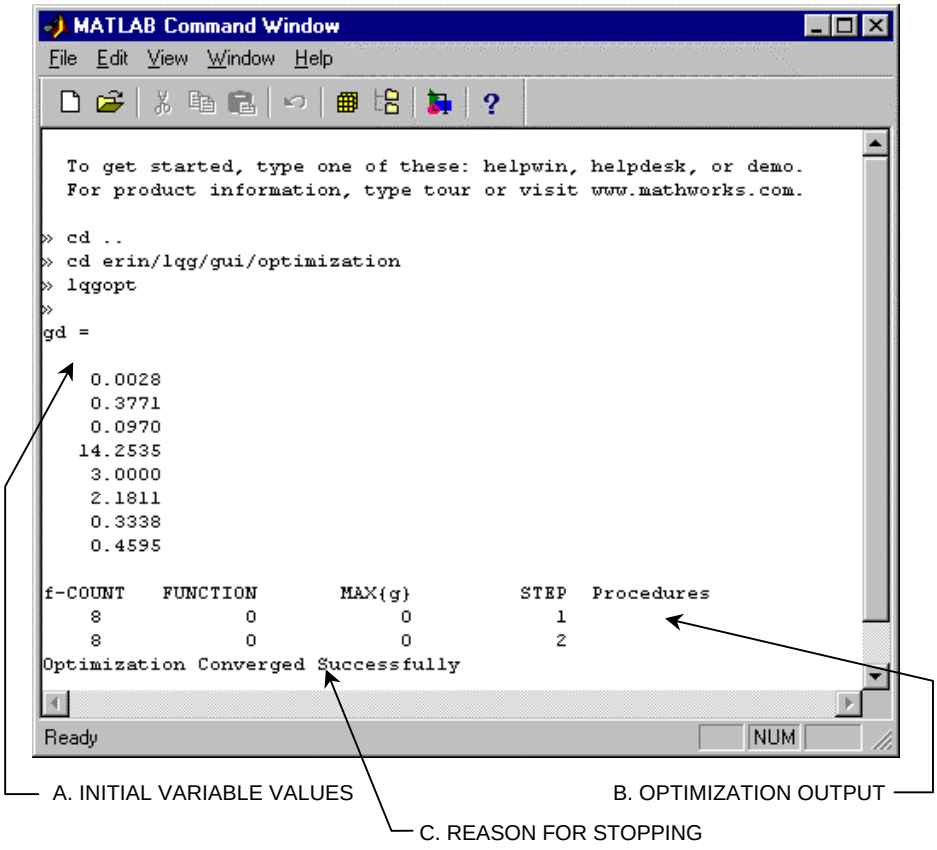


Fig. 18. The fine-tuning interface.



A. INITIAL VALUES OF THE VARIABLES

B. "constr" OUTPUT:

- f-count – ITERATION NUMBER
- function – VALUE OF FUNCTION TO BE MINIMIZED
- max{g} – MAXIMUM CONSTRAINT VALUE
- step – STEP SIZE
- procedures – Infeasible, Hessian modified, etc.

C. INFEASIBLE, SUCCESSFUL CONVERGENCE, MAXIMUM NUMBER OF ITERATIONS

Fig. 19. The fine-tuning Matlab command window.

Figure 20 is a good three-dimensional analog for understanding what the "constr" function is attempting to do. Although the function used in this GUI (summed performance criteria, eight-dimensional) is not nearly as simple as the one pictured, the basic attributes are there, namely the irregularly spaced locations of peaks and wells (minima) of a positive function. The Matlab program begins at the specified initial "point" and randomly searches for these minima. The user may consider the weight vector a marble on the surface below, and, when attempting to minimize, may wish the marble to roll "down" to the smallest (positive) value. The user likely would receive the "infeasible" message were they to input a random start point (weight vector), because chances are that the start point would not fall within the function domain.

The complicated part is when the user adds constraints, limiting the marble not only to the function domain, but also to a new, multi-dimensional constraint domain. To imagine this, draw any shaped region on the same axes; the points at which these two surfaces intersect determine the new hybrid function to be minimized, and the lowest valued point within this region has become the new local minima. The

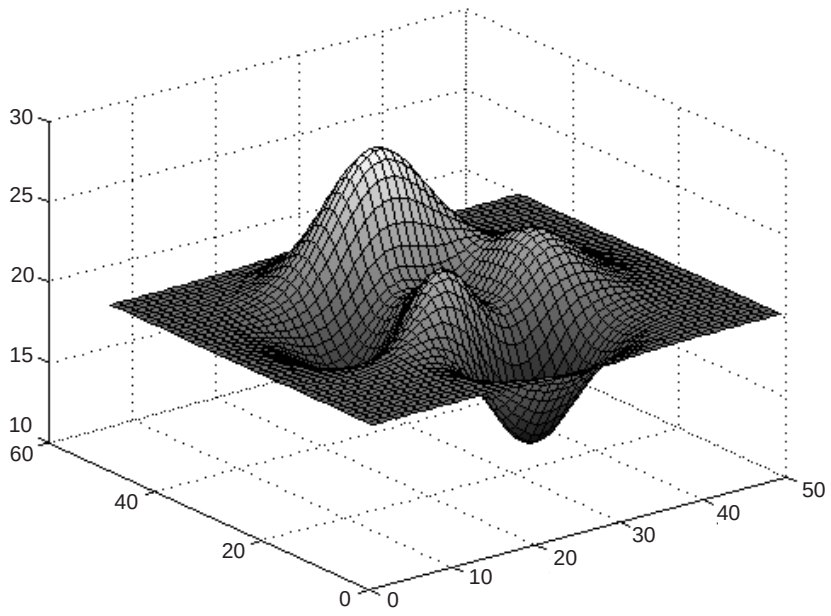


Fig. 20. Function to be minimized.

original function and constraint functions have two unique domains. If these two domains intersect, then the program is active and shall go on to optimize it (assuming that their initial point is close enough to this intersection). But the two domains may not intersect, and this is when Matlab returns the infeasible response.

C. Fine-Tuning Procedure

The following procedure is recommended for tuning the controller using the GUI:

(1) For each variable, verify the starting values by clicking on the "optimization weights" (rather than the "constrain") radio button, and set the weights to zero. Construct the weight vector

$$\begin{bmatrix} \text{Integral} \\ \text{Proportional} \\ \text{Special} \\ \text{Frequency 1} \\ \text{Frequency 2} \\ \text{Frequency 3} \\ \text{Frequency 4} \end{bmatrix}$$

using the last values from the design GUI. (Using arbitrary initial values will most likely result in an "infeasible" result.) Run the optimizer by setting the maximum number of iterations to 1 and by pressing the "optimize" button.

(2) Keep all of the "optimize" radio buttons on, but scale the different variables according to your priorities, i.e., change the weights to nonzero prioritized values. Run this optimization in increments of about 300 iterations or until the system is optimized. Note: If the system does not find the optimal solution on the first run through, be sure to have the weight vector (the push button above the weight-vector display) begin at the previous finish point. To use these values, the "keep" push button must be pushed before the "optimization" push button.

(3) After the system is optimized ("converged successfully" is printed in the Matlab command window), look through the values of the variables and decide which performance values are acceptable and which are not. Of the ones that are not, select the least acceptable. Starting with the least acceptable variable, choose its "constrain" radio button and select a constraint value that is tighter than the current value, but loose enough so that the solution is feasible. If infeasible values are selected, the algorithm will let the user know right away. Refresh the weights if necessary. Again, reset the weight vector to pick up where it left off and run the optimization until it converges successfully.

(4) Repeat the previous step (not forgetting to reset the weights vector before pushing the "optimize" button), constraining each unacceptable variable until the desired output is achieved.

Because it is the gains, K_i, K_p, K_f, and K_e, that are needed to implement a controls design, these values are saved (and replace previous values) after each block of iterations as "gains.mat" in the working directory.

D. DSS-26 Antenna Example

We start with the final design of Section IV and follow the procedure in Section V.C. The steps below correspond to the interfaces in Figs. 21 through 24. The goal is to improve the design by increasing damping of the higher frequency oscillations (the vibrations are visible in the step response) and to improve the rms error due to wind disturbances. Other beneficial results were obtained as a byproduct of our efforts.

(1) Verify the starting values with the weight vector equaling

$$\begin{bmatrix} 12 \\ 25 \\ 0 \\ 5 \\ 0 \\ 10 \\ 0 \end{bmatrix}$$

The result is shown in the interface in Fig. 21, which is identical to the results in the interface in Fig. 17.

(2) Prioritize your goals. In our case (see Fig. 22), we wanted to minimize the rms error, so we assigned the largest weight of 1.0 to this value. Then, we wanted to improve the damping for the high frequencies (and additionally the disturbance transfer function, the settling time, and the maximum value of disturbance); therefore, we selected 0.1 weights for these variables. We were satisfied with bandwidth and rate error; therefore, we weighted these with 0.01. Overshoot was not of concern for our purposes, so we assigned it a weight of 0.001. We chose a maximum of 300 iterations. At the end of these 300 iterations, the system had still not converged, so we started with the weight matrix from the previous iteration and ran the system again for the next 300 iterations. This time it converged after about 100 iterations. The optimization results are visible in Fig. 22, which shows improved values of all variables except overshoot.

(3) Review the performance. We reviewed the performance and decided that the maximum disturbance was unacceptable; therefore, we constrained it to 0.34 at the same time relaxing the weighting of overshoot, settling time, and bandwidth values. Again we set the maximum number of iterations to 300 (see Fig. 22), and the system converged after about 200 evaluations. The results are shown in the interface in Fig. 23, where the constraint was met and rms error, bandwidth, and overshoot were improved. The command transfer function at high frequencies, the disturbance transfer function, and the settling time deteriorated.

(4) Make final corrections. As a last step, we constrained overshoot to 25 percent, leaving everything else as it was in Step (3) (see Fig. 24). The result after completing 300 iterations and another 100 iterations is visible in the interface in Fig. 24, with an overshoot of 23.3 percent and additionally improved disturbance transfer function, settling time, and maximum disturbance.

After completing these four steps, we were satisfied with the results except for bandwidth, which preferably would have been over 2.2 Hz. The next step involved constraining the bandwidth to this value and leaving all other settings as they were. The result was an infeasible request. We next tried decreasing the optimization weights but came up again with an infeasible outcome. At this point, we decided that the results actually were quite "fine-tuned" and called it quits.

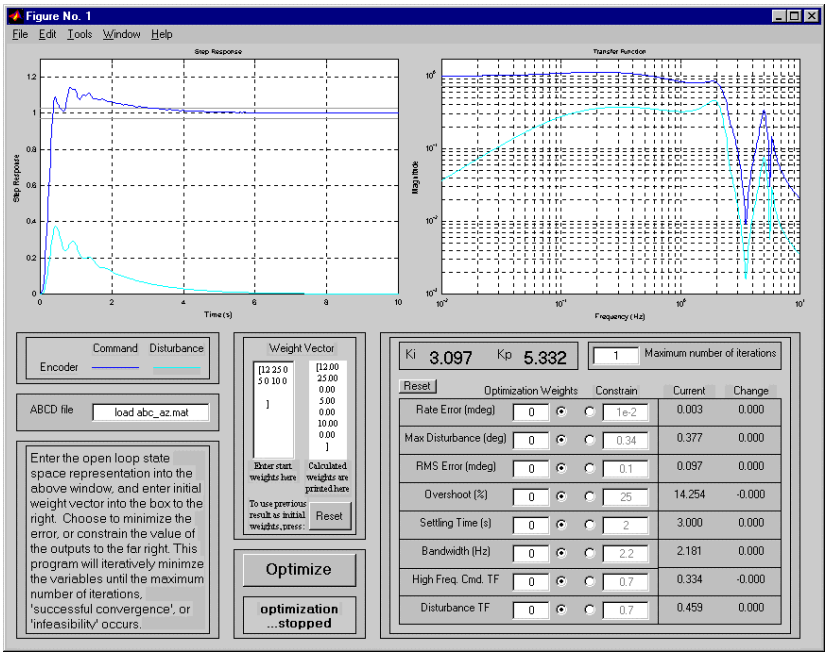


Fig. 21. Fine-tuning interface 1.

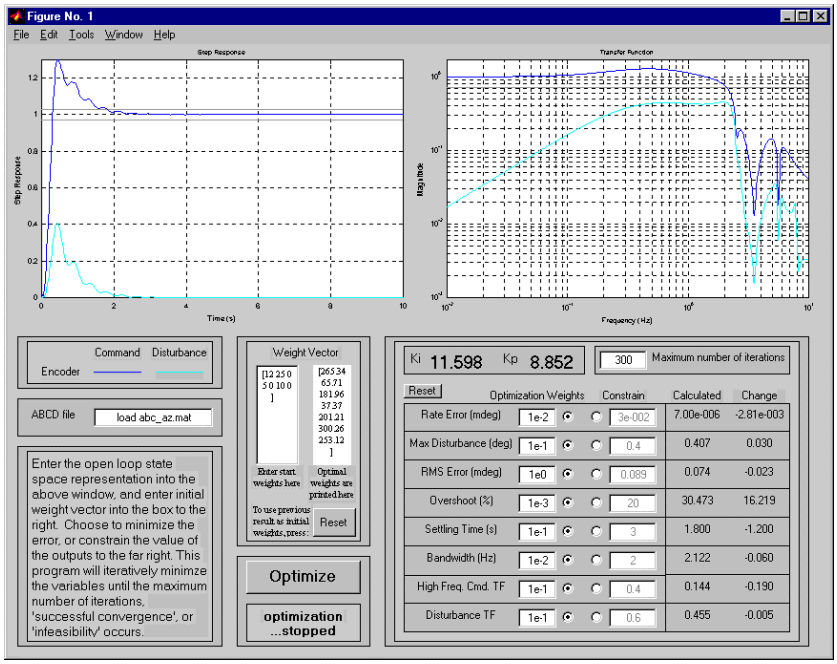


Fig. 22. Fine tuning interface 2.

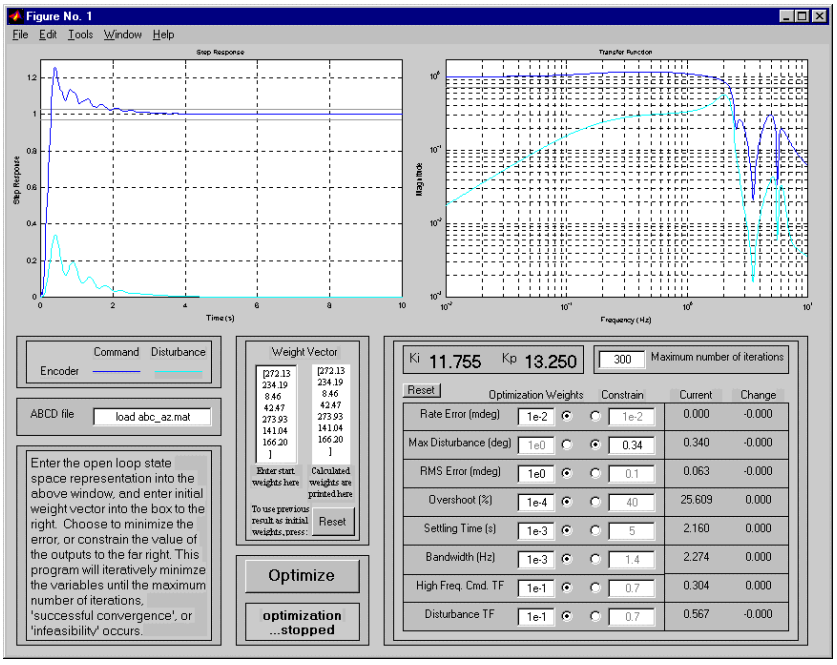


Fig. 23. Fine tuning interface 3.

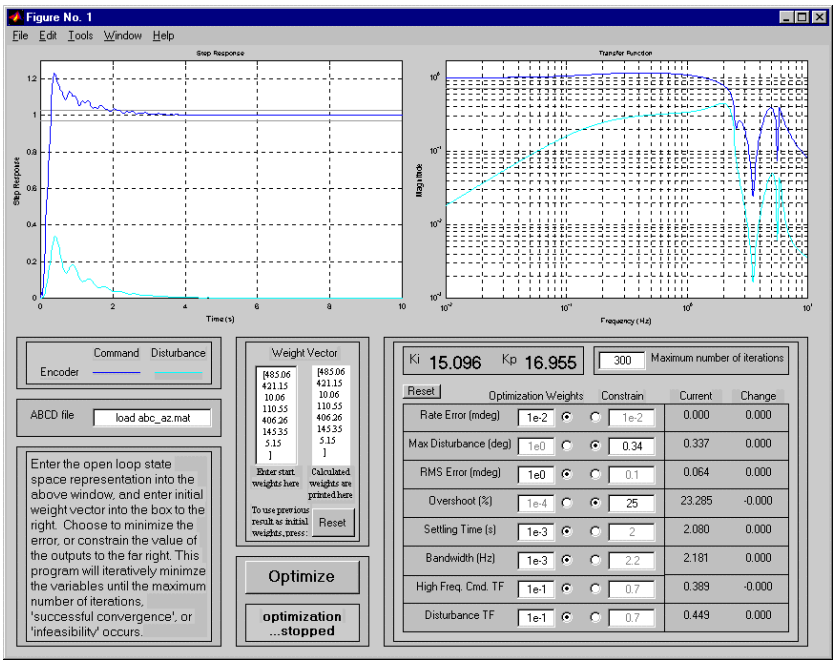


Fig. 24. Fine tuning interface 4.

VI. Conclusions and Future Work

This article describes two types of GUI that help design LQG controllers for the DSN antennas: the LQG controller design GUI and a GUI for the fine-tuning of the LQG controller design. In the *design* GUI, the user simply manipulates controller parameters to observe their impact on the closed-loop system performance (such as overshoot, settling time, bandwidth, rate-offset error, maximum response to step disturbance, and response to wind gusts). A procedure is given that guides the user towards achieving a particular goal. The *fine-tuning* GUI improves the already designed controller by running an optimization procedure. The closed-loop performance parameters, aside from the ones mentioned above, include vibration damping and maximum magnitude of the disturbance transfer function and are either optimized or constrained, so that a desired performance is approached. A procedure is given to guide the user towards achieving performance requirements.

The GUIs were developed for the BWG antennas. Do they work for 70-m antennas? This question has not yet been tested. The main hurdle is the size and configuration of the open-loop model. If the size is larger than 10, or if additional "special" components of the state–space representation resurface, then the GUI interface must be modified. This is not a major technical problem, but if there is a need for the design tool's compatibility with other antennas, the existing GUI shall be tested and, if necessary, modified.

Additionally, the GUIs were developed for the models obtained from system identification. Do they work for the finite-element models? This question is important when the antenna does not exist but is in a design stage. The finite-element models behave significantly differently than do the system-identification models; therefore, this question cannot be answered offhand. Again, the GUI with the finite-element open-loop model shall be tested and, if necessary, modified.

The future development of both design and fine-tuning GUIs shall include a "return" button that would essentially undo the most recent, unsatisfactory design. To further aid the inexperienced user in evaluation and improvement, the step- and transfer-function plots shall include both the initial and the current designs.

References

[1] W. Gawronski, *Dynamics and Control of Structures: A Modal Approach*, New York: Springer-Verlag, 1998.

[2] W. Gawronski and J. A. Mellstrom, "Control and Dynamics of the Deep Space Network Antennas," in *Control and Dynamics Systems*, C. T. Leondes, ed., vol. 63, San Diego, California: Academic Press, pp. 289–412, 1994.

[3] F. W. Fairman, *Linear Control Theory*, Chichester: Wiley, 1998.

[4] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control*, New York: Wiley, 1996.

[5] W. Gawronski, C. Racho, and J. A. Mellstrom, "Application of the LQG and Feedforward Controllers for the DSN Antennas," *IEEE Trans. on Control Systems Technology*, vol. 3, pp. 417–421, 1995.

[6] W. Gawronski, H. G. Ahlstrom, Jr., and A. B. Bernardo, "Design and Performance of the DSS-14 Antenna Controller," *The Telecommunications and Mission Operations Progress Report 42-135, July–September 1998*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–9, November 15, 1998.
http://tmo.jpl.nasa.gov/tmo/progress_report/42-135/135E.pdf

Appendix

Software Requirements, Setup, and Anomalies

In this appendix, we list the requirements for running the GUIs, give a brief description of the files needed, and discuss anomalies encountered when running the programs.

I. Requirements for Running These Applications

The requirements for running these applications are as follows:

(1) Users must have Matlab installed on their computers. All of the requirements for Matlab installation (memory, math coprocessor, hard drive space, etc.) must be met in order for these GUIs to operate correctly. The Matlab optimization and control toolboxes must be installed.

(2) Although the GUIs have not been tested on any version of Matlab below 5.2, it is assumed that any Matlab version that recognizes the "guide" function has the capability to execute the GUI successfully.

II. Setup Procedure

The setup procedure is as follows:

(1) Copy the files listed in Table A-1 to a common Matlab subdirectory.

(2) Make the directory that contains the files listed in Table A-1 the current directory in the Matlab command window, i.e., "cd c:/matlab/lqgguiforbwg."

(3) At the >> prompt, enter "lqg [return]" to open the GUI (enter "lqgopt [return]" for the fine-tuning GUI). From here, follow the methods outlined in the examples.

Table A-1. Files needed to run the design and fine-tuning GUIs.

Design	Fine-tuning	Description
abc_az.mat or abc_el.mat		Discrete state–space representation of the antenna (contains the a, b, c, and d matrices)
Wind_az.mat or Wind_el.mat		Disturbance vector
lqg.m	lqgopt.m	Opens the GUI
lqggui.m	minimx.m, funx.m	Controller algorithm called repeatedly during simulation
naming.m	—	Grabs all user-selected values
balani.m (balan.m, realdiag.m)		Balances the system with poles on imaginary axis (called within)
modal1.m		Finds real block-diagonal modal matrix in increasing order of the imaginary part of the poles of the a matrix
dlqe_my.m (dlqr_my.m)		Discrete linear quadratic estimator design (called within)

III. Anomalies and Restrictions

In the design GUI, a simplified algorithm was implemented to calculate settling time, percentage of overshoot, and maximum disturbance. If the system is stable, these outputs are reliable, but if the system is unstable, the output must be verified with a logic check.

In the fine-tuning GUI, the bandwidth is calculated (intentionally) using a 1.0-Hz rather than a 0.71-Hz limit for the magnitude of the transfer function. The user should not see anything abnormal due to this calculation (except for possibly better results).

If the user selects random initial weights, there is a very good chance that the optimization will be infeasible (again, it is recommended that one pick up the initial weights of the fine-tuning GUI using satisfactory results from the design GUI). Additionally, not every set of weights will converge to a minimum. The program will quit running for one of four reasons:

(1) Infeasible values: With the given weight vector, the requested outputs cannot be attained. In this case, the system will most likely identify infeasible variables before calculating more than 10 function evaluations.

(2) Maximum number of iterations occurs before the minimum is attained: The program will stop and display the weight vector, function value, and maximum constraint as of the last iteration. "Increase options(14)" will appear, alerting the user to "continue running from the last weight vector," or "execute the last weight vector with a larger maximum number of iterations."

(3) Minimum attained: This is the most desirable reason for the program to quit, in which case the function value at the local minimum will be printed in the Matlab command window and the gains should be recorded.

(4) Error occurs: The code will not be executed from the location of the error forward. Because of this, the interface will imply that the program is still running, but in fact it is not, and the plots will not be updated. To determine if the program has encountered an error, the user must frequently refer to the Matlab command window for messages.

Two anomalies come up due to the nature of the "constr" algorithm used in the fine-tuning algorithm. The first is that it cannot capture some outputs that may be of interest to the user, so the value of the minimized function, iteration number, maximum constraint, and step size are printed to the screen. Secondly, the algorithm performs a *random* search for a minimum, and repeatability is not guaranteed.

While running the fine-tuning GUI, a message—"Warning: Q is not symmetric and positive definite"—may appear at the Matlab command window. This message should be ignored since the matrix Q is always positive and symmetric by definition (it is a diagonal matrix with positive weights at the diagonal). The positive definiteness is evaluated through the eigenvalues of Q. IfQ is ill conditioned (it consists of very large and very small values), the eigenvalue routine has a significant numerical error. Since the eigenvalues of Q are not used elsewhere, the run is successful.

Lastly, because of the figure handles, the design and fine-tuning GUIs may not be opened at the same time. If they are, an error message most likely will be displayed, and the plots will not be printed. This is an issue that also will be addressed in a future version.